Interconnecting and Monitoring Heterogeneous Things in IoT Applications

Patient Ntumba, Georgios Bouloukakis & Nikolaos Georgantas*

MiMove Team, Inria Paris, France.
firstname.lastname@inria.fr

Abstract. Internet of Things (IoT) applications incorporate heterogeneous devices that employ different middleware protocols (MQTT, CoAP, WebSocket, etc). In this paper we present an extension of our cross-integration platform which supports the interoperability of IoT devices. In particular, we introduce the *VSB Web Console* which enables the development and monitoring of applications with heterogeneous IoT devices. We showcase our approach using the *Fire Detection* scenario.

Keywords: Internet of things, Middleware Protocols, Interoperability Artifacts, Emergency Response

1 Introduction

Internet of Things (IoT) devices such as smart thermostats, activity trackers, drones, parking sensors, etc, enable developers to create new types of smart applications. Such devices (or *Things*) are introduced and deployed by major tech actors using their own proprietary APIs and protocols. This results in the deployment of highly heterogeneous devices in terms of both hardware and software resources. Additionally, the Things' diversity – i.e., resource-tiny, resource-constrained and resource-rich, prevents the development of applications that rely on a single standard/protocol. Hence, enabling interactions in the IoT requires to deal with the heterogeneity issue.

Existing interoperability efforts are based on *i)* bridging communication protocols; [5,6]; and *ii)* providing common API abstractions[8]. The former approach focuses only on the data and primitive conversion between a specific set of protocols. In the latter approach, developers have to build their application by relying on a single protocol or API. In the context of our work, we have developed the *eVolution Service Bus* (VSB) [7] which enables the interconnection of Things employing different IoT protocols at the middleware layer (i.e., MQTT [4], CoAP [10], WebSocket [9], HTTP, etc). VSB follows the ESB paradigm where a common intermediate bus protocol is used to interact with multiple Things employing middleware-layer protocols or IP-based Gateways hosting IoT devices. To bridge heterogeneous Things with the intermediate bus

* The work is supported by the research associate team ACHOR (inria.fr/en/associate-team/achor) and the EU-funded H2020 project FIESTA-IoT (fiesta-iot.eu)

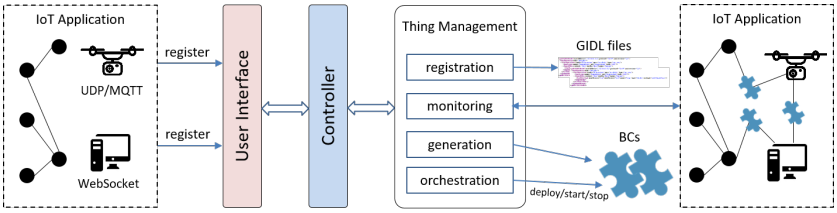

Fig. 1. VSB Web Console Architecture

protocol, we automatically synthesize interoperability software artifacts, the so called *Binding Components* (BCs). BCs perform the mapping between data and primitives of the bridged protocols. VSB is utilized as core component of the H2020 CHOReVOLUTION [2] project and enables interactions in IoT choreographies with heterogeneous devices.

In this paper, we rely on VSB and we introduce the *VSB Web Console*. Our console provides a graphical interface which enables developers to register their services/Things employing middleware IoT protocols. Then, based on their use case scenario, interactions can be enabled between the registered (and possibly heterogeneous) Things. Finally, the resulting application can be monitored through our console. Monitoring options include: message passing and management of the deployed artifacts. To demonstrate our work, we design a use case for fire detection inside a forest. Hence, we register real sensors (temperature sensors), devices (a drone) and services (an estimation service) to our console. Then, to enable the interconnection of heterogeneous devices, we automatically synthesize and deploy interoperability artifacts. Finally, the forest fire detection interactions can be monitored using our console.

In the following section, we provide an architectural overview of our console, as well as details about its implementation. Then, we provide an overview of the forest fire detection scenario. Finally, we demonstrate its implementation in the VSB Web Console – this includes a video representation.

2 System Overview

As depicted in Fig. 1, our console is implemented based on MVC standard. Below we describe its main components which are used to interconnect heterogeneous Things and monitor IoT applications:

User Interface (UI): is a Web interface used by an IoT developer for registering services/Things by providing information such as the: *i)* role (provider, consumer), *ii)* host address, *iii)* employed middleware protocol, *iv)* supported operations and their input/output data. This process results to the creation of the corresponding *Generic Interface Description Language* (GIDL) model. We provide the GIDL models of our use case scenario at `https://goo.gl/Lnzziz`. The UI is also used for interconnecting the registered services/Things using drag and drop actions and automated code generation. Finally, is used to manage and monitor the overall IoT application.

Controller: it processes all actions performed at the UI, which are forwarded to the *Thing Management* component and the corresponding module: *i)* registration, *ii)* generation, *iii)* monitoring and *iv)* orchestration.

To interconnect and monitor two heterogeneous Things, e.g., a drone that sends/receives notifications using the UDP/MQTT protocols; and a media device that receives notifications using the WebSocket protocol, our console operates as follows: the *generation* module binds the drone's and the media device's GIDL models to automatically generate the BC that is responsible to map data and primitives described in their GIDL models. Then, the *orchestration* module interconnects these devices by deploying the BC in the console host node. To monitor the devices and the deployed BC, our console (*monitoring* module) implements a *listener API* which receives published data from the IoT app on a specific port.

Implementation. The VSB Web Console has been implemented using `Java 8` and the GIDL interface is a *metamodel* developed using the *Eclipse Modeling Framework*. Our console can be downloaded through: `https://gitlab.inria.fr/pntumba/vsb-web-console/wikis`. We also provide 2 videos for demonstrating the usage of our console: the 1st one: `https://youtu.be/v7ucoSgbZCI`, demonstrates the process of installing the console. Additionally, we show how a developer can register, interconnect and monitor (heterogeneous or not) Things.

3 Forest Fire Detection Scenario

Continuous monitoring of forests for early fire detection is of primary importance. In 2007, there were more than 80 human losses in Greece and 670,000 acres burned because of fires [3]. In this section, we use our console to showcase the implementation, deployment and monitoring of a fire detection inside a forest. Implementing such a scenario requires the following sensors and devices:

Temperature or **smoke sensors**: these can be deployed inside a forest in order to monitor the forest conditions and push warning notifications. In the context of our demo, we deploy these sensors into a Raspberry Pi and we employ the CoAP protocol to push notifications.

Estimator service: it can be deployed inside a fire department for receiving forest-related notifications. We assume that the estimator service employs the HTTP protocol to receive notifications.

Drone: a PARROT AR DRONE 2.0 located at the fire department – used to inspect the forest-area upon a warning notification. Such a device deploys specific protocols to: *i)* accept pilot-commands through the UDP protocol and *ii)* transmit video stream, location and drone speed to an MQTT broker.

Drone handler: a device handling the drone. We have deployed this device into a BeagleBone Black[1] platform. This platform employs the HTTP protocol to receive commands from the estimator service and locate the corresponding drone into the forest.

Media streaming devices: such devices receive video streaming data by the drone. We separate them in two categories: *i)* a device that employs an MQTT

subscription mechanism; and *ii)* a smartphone that employs the WebSocket protocol.

Creating an IoT application with the above heterogeneous devices is not a trivial task. Initially, the developer has to deploy the services/things, select the proper protocol supporting the constrained sensors/devices or use the already employed protocol by the device (e.g., drone). Hence, the developer has to be aware of multiple APIs and protocols for interconnecting its heterogeneous services/things and create the application.

In this work, we enable developers to register services/Things into the *VSB Web Console* which can be installed in their private or public machine (see Section 2). Then, IoT applications can be created through simple drag and drop actions. Heterogeneous interconnections are solved by generating automatically BCs. Finally, our console provides a monitoring interface to manage BCs (start/stop actions) and message passing detection.

In the 2nd video we show the complete implementation and monitoring of the *Fire Detection* scenario: `https://youtu.be/SJeiqJkBhls`.

4 Conclusion

To facilitate the development of IoT applications, we have developed the *VSB Web Console*. Using our console, developers are able to register services/Things, create IoT apps through drag and drop actions, interconnect heterogeneous Things in an automated manner, and finally monitor the resulting applications.

In future, we intend to provide an API to enable developers accessing the *Controller* directly (not only through the UI). Furthermore, we aim to enable a distributed deployment of BCs.

References

1. BeagleBone Black. `https://beagleboard.org/black`
2. CHOReVOLUTION EU project. `http://www.chorevolution.eu/bin/view/Main/WebHome`
3. Greek forest fires. `https://en.wikipedia.org/wiki/2007_Greek_forest_fires`
4. MQTT version 3.1.1. `https://goo.gl/1WdTPZ`
5. Ponte. `http://www.eclipse.org/proposals/technology.ponte/`
6. Al-Fuqaha, A., et al.: Toward better horizontal integration among iot services. IEEE Communications Magazine (2015)
7. Bouloukakis, G., et al.: Integration of Heterogeneous Services and Things into Choreographies. In: ICSOC. Banff, Alberta, Canada
8. Cherrier, S., et al.: D-lite: Building internet of things choreographies. arXiv (2016)
9. Fette, I., Melnikov, A.: The WebSocket Protocol. Tech. rep. (2011)
10. Shelby, Z., Hartke, K.: The Constrained Application Protocol (CoAP). Tech. rep. (2014)